COEN6311 Deliverable 3: Implementation, Deployment GroupSuper

1st Jun Huang
Concordia University
Montreal, Canada
jun.huang@mail.concordia.ca

2nd Ding Li
Concordia University
Montreal, Canada
ding.li@mail.concordia.ca

3rd Zerui Wang
Concordia University
Montreal, Canada
zerui.wang@mail.concordia.ca

I. Introduction

This is the third deliverable report of COEN6311 course group project. This report articulates the system architecture designs and presents the current progress the group has done so far.

The report consists of two parts. The first part will first introduces the system architecture designs and the software engineering process which comes up with those designs. And then the reversion of the system will be presented. And finally, more details of the Layered MVC pattern will be discussed.

The second part illustrates the software metrics and granularity of components.

II. Part I

A. Identify and articulate what are your architectural designs and the associated software engineering process.

This section will First discuss the design of both external and internal architecture and the reason of why they are design in this way. After that, the process of how the designs are designed will be presented.

*1) **Architectural Designs**:* As described in the previous deliveries, the software external and internal architectures are designed based on requirements that have been specified in detail. Fig. 1 and Fig. 3[1] demonstrate the logical view and physical view of the system by block diagram respectively.

With the external architecture design, the users are able to register, log in, update user information, search paper, evaluate and share paper through the web browser. User activities which involve paper operations can be captured and accessed through the ICDE API for the system itself and also for third party applications to derive information about users' preferred papers and their browsing habits. The above data is available in the user search engine database and the ICDE database respectively.

With the internal architecture design, the system can be divided into three logical stacked layers with implementing

[1]The internal architecture has been redesigned due to some reversions, so it looks different from the previous one.

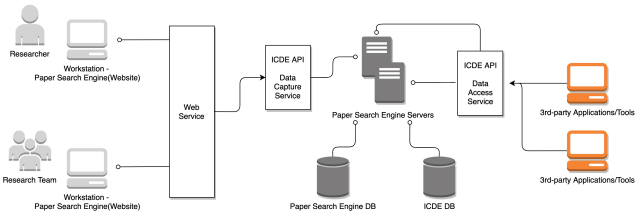

Fig. 1. Block Diagram of the External System Arch. in Physical View

the MVC pattern[2] at the same time. Different layer focuses on the different logical data type as can be seen in Fig. 2.

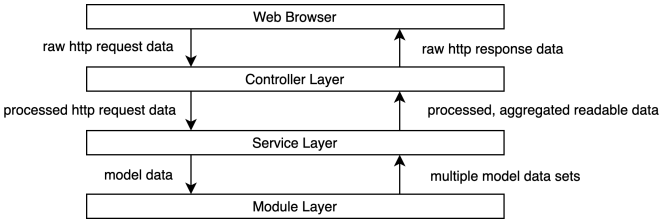

Fig. 2. The Datatype in the Layered System Architecture

1 **The controller layer** contains the controllers which are responsible for receiving and responding to the HTTP requests sent from users' browsers. This layer focuses on the communication between the system and the browsers. Each controller accepts one business aspect of requests and calls the corresponding service methods to retrieve the demanding data, and warps the date to unified standard HTTP responses which will return to the browsers.

2 **The service layer**[3] contains the services which are responsible for preparing and organizing the system data. This layer focuses on responsing the call from controllers and calls the corresponding module methods to manipulate the data. Each service should be able to handle the data in bi-direction. For instance, one service receives the

[2]The MVC pattern introduced in the project is not exactly the same MVC pattern presented by the textbook [1], this will be discuss at section II-C.

[3]This layer is also well known as bussiness layer/logical layer.

raw data sent from the controller and do some calculation before passing the data to the persistence module. Or service calls methods from several persistence modules and aggregates those different data sets into one set then return it to the controller.

3 **The module layer** contains two parts of the system,

- **Part one**[4] is all about the data persistence modules which are responsible for data persistence and data query. This layer part focuses on the communication between the system and the databases. Each module takes the corresponding data sent from services and saves them into the databases, or queries data for the services.
- **Part two** contains all the common modules or utilities that can be reused in any part of the system. For instance, the common superclasses or string formating handler or customized file reader etc.

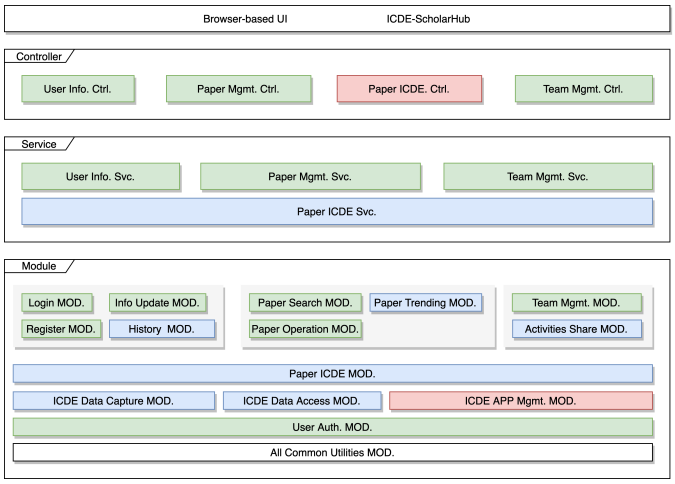

Fig. 3. Block Diagram of the Internal System Arch[5][6]. in Logical View

*2) **Software Engineering Process for Designing the Architecture**:* For how the system external architecture was designed is quite simple: the project is designed to be a web-based application, hence a traditional B/S architecture is used to describe the physical component of the system.

As for the internal system architecture design, the activity diagram of the process is shown in Fig. 4. As discussed before, each system layer was divided by what data type it handles. Activities with the blue color are the main activities that decide how the process is performed. And the objects with the green color are the outcome product after certain activities. After the process

[4]This part is also well known as the persistence layer.

[5]The system architecture is different from the one which submited in the previous deliveries.

[6]Components in the green box are currently shippable, components in the blue box are on the todo list or under construction. And the components in the red box will not be implemented in the scope of the class project.

B. Indicate if any revision to your architectural design is necessary. If the answer is Yes, please explain what revision and the reason.

There are two reversions introduced into the system architecture during the past two sprints. All reversions were done at the designed level before the actual code was implemented, hence the effects of the system changing are small and smooth.

The following features were cancelled:

- **Paper Upload & Download Module:** At the initial system requirement, the system should be able to upload and download the pdf file of the paper. But due to the copyright issues, the system could not maintain the pdf file database so those two features were removed. Instead, the system only returns an URL that leads to the original paper page.
- **WebSocket Module:** At the initial system requirement, team activities were designed to be updated in real-time, not just polling requests by the front-end javascript so the system will maintain a WebSocket server for instant notification. But after the latest evaluation of the user experience of the system, this real-time notification requirement is not that important. So the module was removed to reduce the complexity of the system.

C. Design decisions between MVC architecture pattern and layered architecture pattern.

For the internal system architecture design of this project, **a combination of MVC architecture pattern and layered architecture pattern has been applied.**

As can be referred to [2], there are various versions of how to interpret the Model-View-Controller over its original philosophy, the disputable part is “Can the model layer be communicating with the view layer?”.

The version introduced by the textbook [1] put the answer yes on the question, which is quite disputable. Because the mission of the model by its definition is all about managing the data, it should never put effort into how many views are subscribing to which models and even send notifications to the views.

If the answer is no, which makes sense, the MVC is actually a layered structure [3], [4] since the model layer should not and should never directly communicate with the view layer. In this report, this interpretation is marked as “Layered MVC Pattern”and it is what this project was following.

*1) **Adopt the Layered MVC Pattern**:* how it match the internal system architecture

Fig. 5 presents the project architecture from an MVC perspective view and describes where are the scopes for Model, View, and Controller.

*2) **Pros and Cons of Layered MVC**:* how it affect the system and the engineering process

Advantages of MVC:

- Separation of the code of front-end and back-end.
- Reuse of the components.
- Easy to maintain.

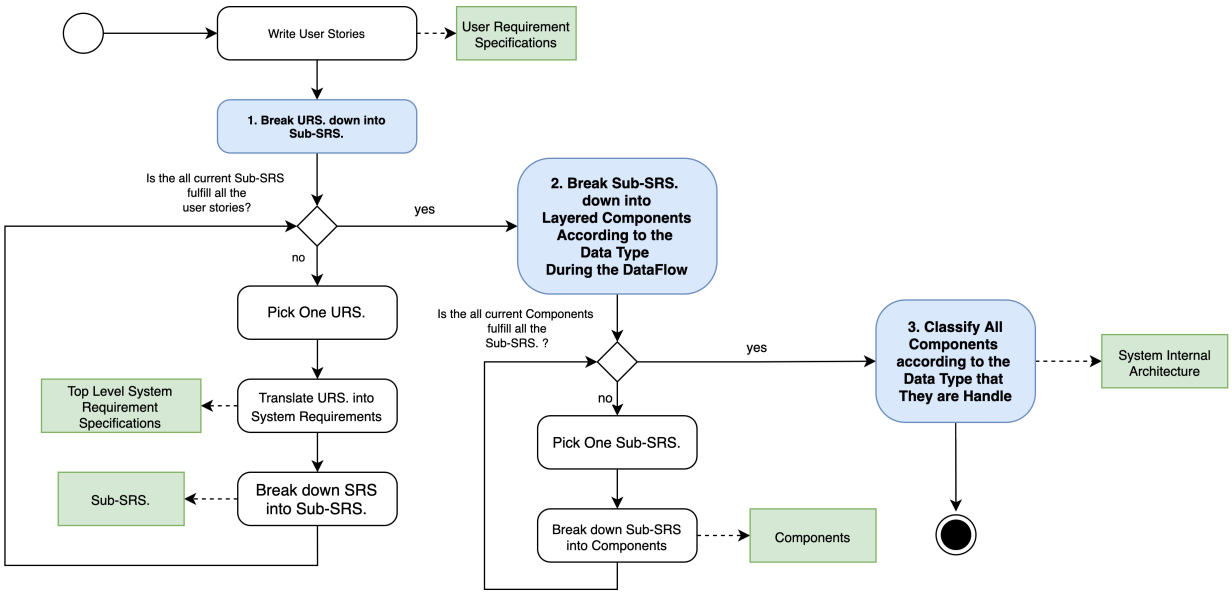

Fig. 4. Activity Diagram of the Process for Designing the Internal System Architecture

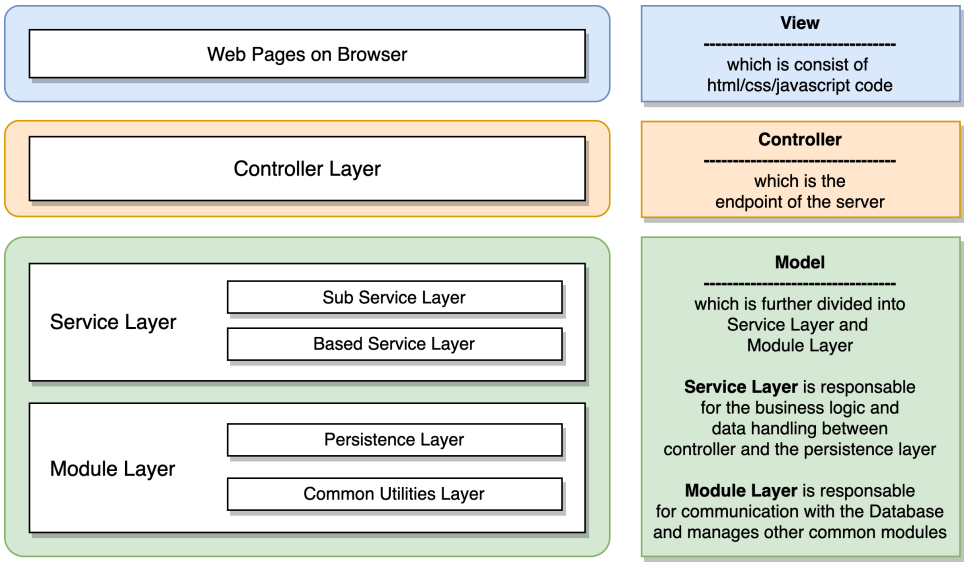

Fig. 5. Matching the Layered MVC into the internal system architecture

- Easy to test and varify the code independently.

Advantages of Layering:

- Developers can focus on the layers where they were more good at.
- Layering the system can easily perform the data tracing during the test.
- Code structure is clear for reading, debuging, and for testing.

Disadvantages of both:

- The structure is relatively complex for a samall project. For this project, the controller layer and the service layer can be merged to one layer.

Overall, the MVC pattern and the Layer architecture pattern share the same idea: MVC pattern makes each part focus on their own jobs while further layering each part is more sensible for component separation and management.

A layered architecture abstracts the overall view of the system. Therefore, the project takes the advantage of this for both patterns.

III. Part II

Table I shows the code granularity that the project has implemented so far in terms of classes/objects, packages, libraries, frameworks and platforms.

TABLE I
CODE GRANULARITY OF THE CURRENT STATE OF THE PROJECT

Task Name	Tables Count	Code Level	Components Count	Loc	Test Cases Count	Endpoints Count
SRS 1.1	1	Service	1	70	4	
User		Controller	1	7	4	4
Information		Persistence[7]	1	41	0	
Management		Vue [8]	2	318	0	2
SRS 1.2	1	Service	3	30	2	
Login		Controller	1	4	2	2
Logout		Persistence	0	0	0	
		Vue	1	118	0	1
SRS 2.2	2	Service	1	176	5	
Paper		Controller	1	4	5	5
Operation		Persistence	1	31	0	
		Vue	1	531	0	1
SRS 3.1	0	Service	0	0	0	
Paper		Controller	0	0	0	0
Search		Persistence	0	0	0	
Engine		Vue	1	531	0	2
SRS 6.1	2	Service	1	200	6	
Team		Controller	1	11	6	6
Information		Persistence	0	0	0	
Management		Vue	1	468	0	2

REFERENCES

[1] Sommerville I. Software Engineering GE[M]. Pearson Australia Pty Limited, 2016.

[2] Aihara D S. Study About the Relationship Between the Model-View-Controller Pattern and Usabiltity[J]. 2009.

[3] D. Zhang, Z. Wei and Y. Yang, "Research on Lightweight MVC Framework Based on Spring MVC and Mybatis," 2013 Sixth International Symposium on Computational Intelligence and Design, 2013, pp. 350-353, doi: 10.1109/ISCID.2013.94.

[4] Morse S F, Anderson C L. Introducing application design and software engineering principles in introductory cs courses: model-view-controller java application framework[J]. Journal of Computing Sciences in Colleges, 2004, 20(2): 190-201.

[7]The persistence layer.

[8]The view layer which use Vue Framework to implement.